FrameFire: Enabling Efficient Spiking Neural Network Inference for Video Segmentation

Chen, Qinyu ; Sun, Congyi ; Gao, Chang ; Fang, Xinyuan ; Luan, Haitao

Abstract: Fast video recognition is essential for real-time scenarios, e.g., autonomous driving. However, applying existing Deep Neural Networks (DNNs) to individual high-resolution images is expensive due to large model sizes. Spiking Neural Networks (SNNs) are developed as a promising alternative to DNNs due to their more realistic brain-inspired computing models. SNNs have sparse neuron firing over time, i.e., spatio-temporal sparsity; thus they are useful to enable energy-efficient computation. However, exploiting the spatio-temporal sparsity of SNNs in hardware leads to unpredictable and unbalanced workloads, degrading energy efficiency. In this work, we, therefore, propose an SNN accelerator called FrameFire for efficient video processing. We introduce a Keyframe-dominated Workload Balance Schedule (KWBS) method. It accelerates the image recognition network with sparse keyframes, then records and analyzes the current workload distribution on hardware to facilitate scheduling workloads in subsequent frames. FrameFire is implemented on a Xilinx XC7Z035 FPGA and verified by video segmentation tasks. The results show that the throughput is improved by 1.7× with the KWBS method. FrameFire achieved 1.04 KFPS throughput and 1.15 mJ/frame recognition energy.

DOI: https://doi.org/10.1109/aicas57966.2023.10168660

FrameFire: Enabling Efficient Spiking Neural Network Inference for Video Segmentation

Qinyu Chen, Congyi Sun, Chang Gao, Xinyuan Fang and Haitao Luan

Abstract—**Fast video recognition is essential for real-time scenarios, e.g., autonomous driving. However, applying existing Deep Neural Networks (DNNs) to individual high-resolution images is expensive due to large model sizes. Spiking Neural Networks (SNNs) are developed as a promising alternative to DNNs due to their more realistic brain-inspired computing models. SNNs have sparse neuron firing over time, i.e., spatio-temporal sparsity; thus they are useful to enable energy-efficient computation. However, exploiting the spatio-temporal sparsity of SNNs in hardware leads to unpredictable and unbalanced workloads, degrading energy efficiency. In this work, we, therefore, propose an SNN accelerator called FrameFire for efficient video processing. We introduce a Keyframe-dominated Workload Balance Schedule (KWBS) method. It accelerates the image recognition network with sparse keyframes, then records and analyzes the current workload distribution on hardware to facilitate scheduling workloads in subsequent frames. FrameFire is implemented on a Xilinx XC7Z035 FPGA and verified by video segmentation tasks. The results show that the throughput is improved by 1.7× with the KWBS method. FrameFire achieved 1.04 KFPS throughput and 1.15 mJ/frame recognition energy.**

Index Terms—**Workload balance, SNNs, VLSI.**

I. Introduction

In recent years, Deep Neural Networks (DNNs) have achieved state-of-the-art accuracy in image recognition tasks [1]–[3]. DNNs are also useful in tasks that involve video processing, such as autonomous driving. These tasks are more challenging since they require real-time processing of input video frames while computing DNNs simultaneously. Thus, fast neural network inference is critical for video processing tasks. Generally, natural videos exist with varying degrees of spatio-temporal redundancy. Various efforts in DNNs have been made to make the process more efficient by exploiting such data redundancies. For example, Taylor expansions are used to produce a compressed DNN by replacing the ReLU function with dynamically updating masks [4]. However, such continuous-valued networks usually have tremendous parameters, prohibitively expensive when deployed on mobile devices, especially in always-on scenarios.

Another approach is to provide efficiency is to employ Spiking Neural Networks (SNNs). Compared to traditional DNNs, SNNs use an event-based model that mimics the spiking behavior of biologic neurons. Therefore, SNNs promise to achieve better performance with lower complexity, especially in applications with temporal sequential data streams [5]. SNNs are stateful and thus suitable for processing natural videos. In a natural video, frames are highly correlated. The network state of SNNs, i.e., the neuron's membrane potential, can keep potentially useful initialized information for the subsequent frames to speed up the convergence. However, it might cause decreased accuracy when outdated information from the previous frame is carried over. To capitalize on the useful initialization, we previously proposed an adaptable interval reset of the network state [6], which can achieve 35x speedup with acceptable accuracy loss.

From a hardware perspective, information travels between neurons in the form of binary spikes in SNNs, resulting in an advantage of higher energy efficiency. Previous research on specialized SNN hardware design, such as large-scale systems (e.g. Loihi [7] and TrueNorth [8]) and low power accelerators for resource-constrained applications [9]–[13] have demonstrated that event-driven SNNs can be efficiently processed. The energy efficiency primarily benefits from the spatio-temporal sparsity and replacement of multiply-accumulate operations by addition. However, the spatio-temporal sparsity induces an unpredictable and dynamic workload, which is challenging to be exploited efficiently in hardware.

In this work, we propose an SNN accelerator, FrameFire, exploring the workload balance for fast video processing. The main contributions are:

- A Keyframe-dominated Workload Balance Schedule (KWBS) method is developed to facilitate scheduling workloads in consecutive frames, achieving 1.7x throughput.
- An SNN hardware accelerator, FrameFire, for efficient video processing with the KWBS method is proposed.
- The proposed accelerator FrameFire is implemented on a Xilinx XC7Z035 FPGA and verified by video segmentation tasks. The results show 1.04 KFPS at 200 MHz.

II. Motivation

SNNs have intrinsically event-driven workloads since the update of membrane potentials is triggered by spikes. As exemplified in Fig. 1, a large number of neurons barely fire over the timesteps (dark blue), inducing considerable sparsity. Connections with zero inputs can be skipped to save computation and memory access. Although exploiting spatio-temporal sparsity can reduce the memory footprint

This work was conducted when Qinyu Chen was with the Institute of Photonic Chips, University of Shanghai for Science and Technology, Shanghai, China, (now she is with the Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland). Congyi Sun is with Nanjing University, China. Chang Gao is with Department of Microelectronics, Delft University of Technology, Netherlands. Xinyuan Fang and Haitao Luan (corresponding author) are with the Institute of Photonic Chips, University of Shanghai for Science and Technology, Shanghai, China.

This work was partially supported by The National Key Research and Development Program of China (Grant No. 2022YFB2804301)

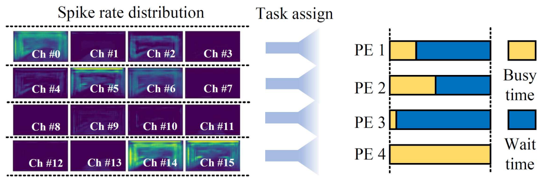


Fig. 1. The load unbalance can be observed in perspective of different input channels within the feature maps (in a spiking convolutional layer).

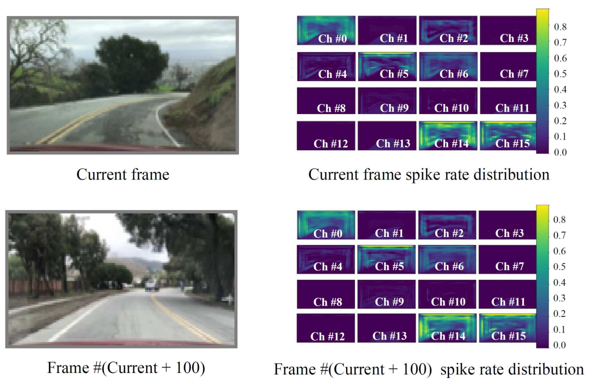
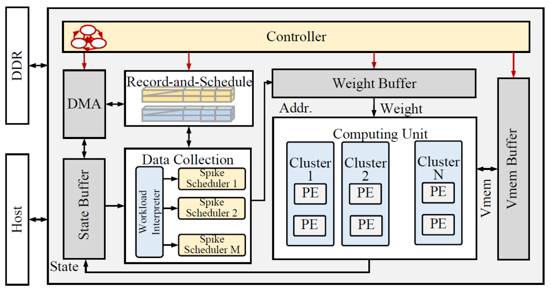
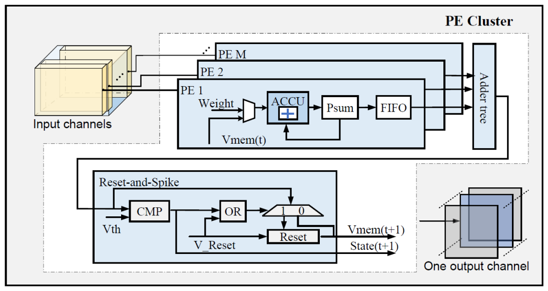


Fig. 2. Here we visualize the spike rate distribution of current frame and frame #(current+100) on the penultimate layer of an image segmentation network.



(a)



(b)

Fig. 3. The proposed system: (a) overall architecture; (b) architecture of the PE Cluster.

and latency, general-purpose processors cannot meet these requirements with high efficiency. Hence, a specialized event-driven architecture targeting video processing is necessary. To realize high parallelism, operations in SNNs must be operated by multiple processing elements (PEs). In this case, PE having the most workload will become the bottleneck of performance, reducing the throughput. Fig. 1 shows the significant load unbalance among different PEs when observing the spike rate distribution of different channels in a spiking layer, most of the computing resources (PEs) are wasted, thus reducing the hardware efficiency. As shown in Fig. 2, the related information of two separate frames is visualized when processing a driving video. It demonstrates that the spike rate distributions of these two frames are quite similar, indicating a similar workload distribution.

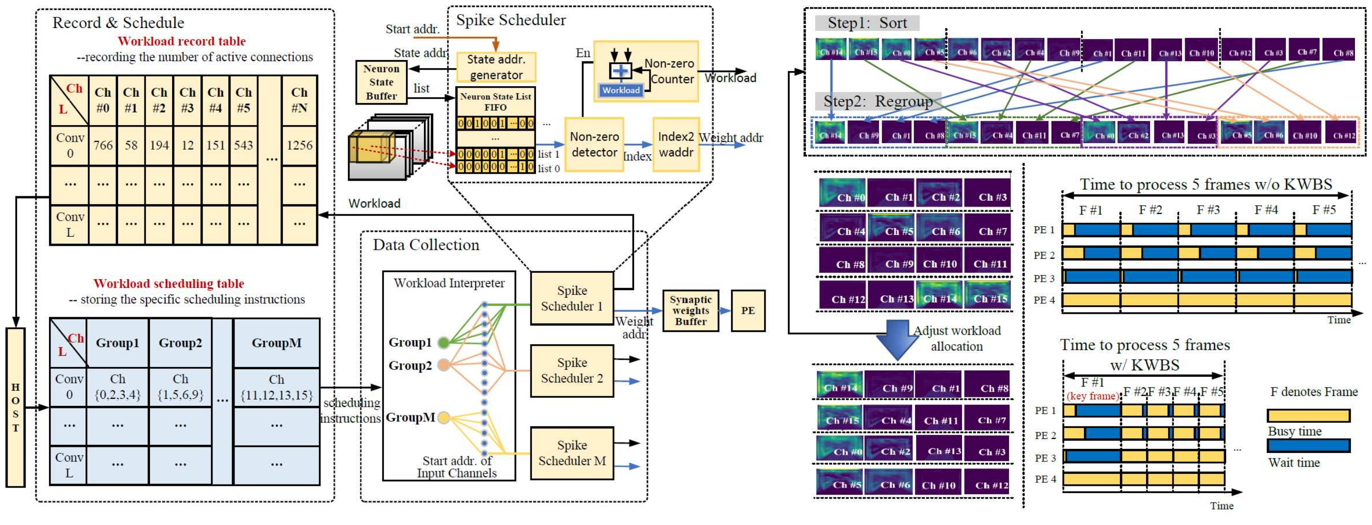
III. Architecture Design and Data Processing

To exploit workload balance and facilitate video processing, the Keyframe-dominated Workload Balance Schedule (K-WBS) method is introduced. The workload distribution when processing networks with sparse keyframes will be recorded and transferred to the host for load balancing scheduling. In this way, the workload for subsequent frames can be easily scheduled due to the similar workloads between nearby frames. Besides, a FPGA-based SNN accelerator is designed to accelerate video processing, supporting KWBS method.

A. *FrameFire: Architecture Overview*

Overall architecture. The block diagram in Fig. 3(a) gives an overview of the proposed accelerator. FrameFire consists of a controller, on-chip buffers, a workload record-and-schedule unit, a data collection unit, and a computing unit. *On-chip buffers* include a neuron state buffer, a membrane potential buffer, and weight buffers. The neuron state buffer prepares the input neuron states consumed by the computing unit and stores the generated output states. The membrane potential buffer stores all intermediate membrane potentials to reduce the latency and power consumption brought by off-chip memory read and write. The data collection unit and workload record-and-schedule unit are the main modules for scheduling the workloads. *The data collection unit* with a workload interpreter and multiple spike schedulers is introduced to check the spikes and arrange the inputs for PEs. *The workload record-and-schedule unit* can communicate with the host and data collection unit to provide the scheduling information. *The computing unit* takes charge of most computation tasks in the SNN model, i.e., the update of membrane potentials. It consists of several PE clusters. *The controller* manages the overall execution, which updates the state of the accelerator and decodes the information fetched from the host.

PE cluster architecture. Fig. 3(b) shows the architecture of PE cluster. It primarily comprises multiple PEs, an adder tree, and a reset-and-spike unit. Each PE cluster is responsible to compute a slice of the output channels independently. PE is the basic computation unit and also the grain of workload scheduling. It receives a slice of input channels with a partial weight matrix to produce partial sums of membrane potentials. The adder tree collects the partial sums in the same position of outputs from all PEs within a cluster and generate temporary membrane potentials. In the reset-and-spike unit, whether a neuron fires or not and whether the temporary membrane potential of a neuron need reset or not, are determined. There



(a) **(b)**

Fig. 4. Demonstration of KWBS method, (a) dataflow targeting workload scheduling, (b) frame processing flow without and with KWBS method.

is a comparator in each reset-and-spike unit for comparing the temporary membrane potential and the threshold V_{th}. If the membrane potential is higher than V_{th}, a spike is produced. As for the reset, there are two different situations. One is the regular reset, where the membrane potentials are reset by subtracting V_{th} from the membrane potentials. Another is the global interval reset, after configurable interval frames, the membrane potentials of the entire network might need reset to zeros. When the configuration content V_Reset is set to 1, the global interval reset takes charge; otherwise, the regular reset works. After that, the updated neuron state and membrane potential are saved back to buffers.

B. Key-frame-dominated Workload Balance Schedule (KWBS)

Workload unbalance arises when the number of computation cycles consumed by each PE is different, due to the varying degree of sparsity. The PE which executes the most zero-involved computations will first complete the task, and has to wait until the other PEs finish the computations; thus the throughput may drop substantially. To address this problem, efforts should be made to deal with intrinsically event-driven workloads. Hence, a Keyframe-dominated Workload Balance Schedule (KWBS) method applied to the FrameFire architecture is proposed, mainly executed in the record-and-schedule unit and data collection unit.

The record-and-schedule unit. As shown in Fig. 4(a), the record-and-schedule unit includes two tables: the workload record table and workload schedule table. The workload record table records the number of active connections in all channels across the layers, preparing to schedule workloads of nearby frames. The workload schedule table stores the specific scheduling instructions transferred from the host.

The data collection unit. The data collection unit is composed of a workload interpreter and multiple spike schedulers. The workload interpreter interprets the specific scheduling instructions transferred from the workload schedule table to several groups of the input start addresses. Each group of the input start addresses will be assigned to a specific spike scheduler. The assigned input start addresses determine the workload of PEs connected with the corresponding spike scheduler. The spike scheduler can undertake two main functions simultaneously: 1) detecting the neuron that produces a spike and generating the memory address of the corresponding weight, thus filtering out the active connections, 2) counting the workload of the current input channel, i.e., the number of active connections in each input channel, and transferring the workload numbers to the workload record table in the workload record and schedule unit. The spike scheduler primarily consists of a state address generator, a neuron state FIFO, a non-zero detector, an index2addr unit, and a non-zero counter. The state address generator generates the address of the neuron state list according to different start addresses. The neuron state lists are loaded to the local FIFO from the neuron state buffer, and then dispatched to the non-zero detector to get the index of active connections. The index will be decoded to weight address by the index2addr unit. In this way, the weights will be directly loaded from weight buffers to PEs for computation. Meanwhile, once a non-zero state is detected, a positive enable signal will be generated and sent to the non-zero counter to accumulate the workload. Each time the workload of an input channel is obtained, it will be sent to the workload record table.

The Procedure of applying KWBS method. Processing videos with continuous frames can be divided into three steps. First, the SNN model is mapped to the accelerator with a keyframe. The workload distribution across input channels will be calculated by the spike schedulers and recorded in the workload record table. Second, the workload distribution of the keyframe in the workload record table will be sent to the host. The *channel-wisely* balanced workload allocation plan

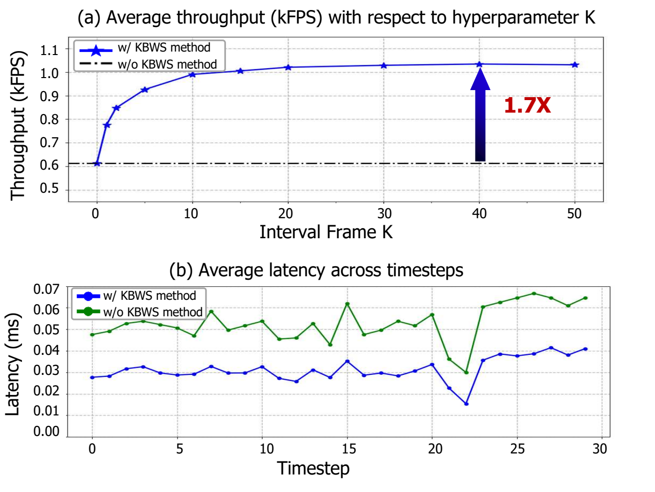


Fig. 5. Experimental results when running across the video recordings, (a) Average throughput (kFPS) with respect to hyperparameter K, (b) latency across the timesteps (K=40).

TABLE I
XC7Z035 FPGA RESOURCE UTILIZATION OF SKYDIVER

Metrics	LUT	FF	DSP	BRAM
Avaliable	171900	343800	900	500
Used	41930	16237	0	128
Percentage	24.39%	4.72%	0%	25.60%

is obtained by the host and sent back to the work schedule table. As shown in Fig. 4(b), the input channels are sorted according to the workload, and then regrouped to M groups. Third, the workload allocation will refer to the scheduled workload instructions when processing the interval frames between every two keyframes. The keyframe is randomly selected from the video recordings of every K frames. Because the selected K is usually big, the communication overhead such as latency and power consumption between the host and the accelerator can be ignored. Fig. 4(b) also demonstrates the frame processing flow without and with KWBS method. The latency is reduced when the KWBS method is applied, since the computation cycles of interval frames are much less than the keyframe due to the adjusted workload allocation.

IV. Experimental Results

In this work, we study the task of detecting lanes in driving videos. As for the spiking segmentation model, we use an encoder-decoder architecture network to realize the end-to-end pixel-wise prediction. The model is tested on the dataset from MLND-Capstone project[1], achieving 97.10% accuracy. For converting the DNN we use an open-source SNN toolbox[2] which implements the conversion method described in [14].

The most important feature of FrameFire is its ability to accelerate video processing efficiently. To verify and analyze the performance, we investigate the effects of adding the KBWS method. The SNN model is mapped to the accelerator

[1]https://github.com/mvirgo/MLND-Capstone/
[2]https://snntoolbox.readthedocs.io/

TABLE II
COMPARISON WITH PREVIOUS WORKS

Metrics	TCAS-II'21 [15]	ICCAD'20 [16]	ASSCC'19 [17]	This work
Platform	VC707	XCZU9EG	XC7VX690T	XC7Z035
Network	MLP	MLP/CNN[1]	MLP[2]	CNN[3]
Dataset	MNIST	MNIST	MNIST	MLND-stone
Task	image classifi.	image classifi.	image classifi.	video seg.
Freq. (MHz)	100	125	-	200
Power (W)	1.6	4.5	0.7	1.2
Pred. Energy (mJ/frame)	5.04	2.34/33.84	0.77	1.15
KFPS	0.32	1.92/0.13	0.91	1.04
Throughput (GSOp/s)	–	–	0.73	23.2
Efficiency (GSOp/s/W)	–	–	0.95	19.3

[1] Classification network with 784-500-500-10 and 28x28-32C3-P2-32C3-P2-256-10 for MNIST.
[2] Classification network with 784-512-384-10 for MNIST.
[3] Segmentation network with 160x80x3-8C3-16C3-32C3-32TC3-16TC3-1TC3-160x80x1, C and TC denote the convolutional layer and transposed convolutional layer, respectively.

with K values ranging from 0 (recovering the baseline) to 50. When evaluating the performance of the driving videos (Fig. 5(a)), the frames per second (FPS) is improved from 0.61 KFPS to 1.04 KFPS when K increases to 40, and slightly drops when K increases to 50, implying that the throughput will be affected if the K value is too high, it is because that the outdated information of workload allocation might negatively affect the performance. The selection of K values depends on the speed of content change over the video frames and the specific application scenarios. Fig. 5(b) also gives the computing latency when K equals 40, it shows that at each timestep, the KBWS method can well schedule the workload regardless of the dynamic nature of SNNs.

The proposed FrameFire accelerator is synthesized and implemented on XC7Z035 FPGA running at 200 MHz. The host is responsible for sending data into the programmable logic part and collecting the results. The resource utilization is summarized in Table I. Table II presents the results of this work and prior state-of-the-art processors. This work achieves 1.04 KFPS and 1.15 mJ/image prediction energy when processing the video segmentation network. Compared with previous SNN accelerators [15]–[17] mainly focusing on processing image tasks, our work is specifically designed to optimize the video processing, and can process larger networks with competitive prediction throughput.

V. Conclusion

In this paper, we proposed a high-throughput SNN accelerator, FrameFire, for efficient video processing. The proposed KWBS method can be applied to FrameFire to alleviate the workload unbalance. FrameFire was implemented on a Xilinx XC7Z035 FPGA and verified by video segmentation tasks. The results show that the FPS was improved by 1.7× to 1.04 KFPS.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2015, pp. 3431–3440.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 779–788.

[4] B. Rueckauer and S.-C. Liu, “Contraction of dynamically masked deep neural networks for efficient video processing,” *IEEE Trans. Circuits Syst. Video Technol*, 2021.

[5] S. Sen, S. Venkataramani, and A. Raghunathan, “Approximate computing for spiking neural networks,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, pp. 193–198.

[6] Q. Chen, B. Rueckauer, L. Li, D. Tobi, and S. Liu, “Reducing latency in a converted spiking video segmentation network,” in *Proc. IEEE Int. Symp. Circuits Syst.(ISCAS)*, 2021, pp. 1–5.

[7] M. Davies *et al.*, “Loihi: A neuromorphic manycore processor with on-chip learning,” *IEEE Micro*, pp. 82–99, 2018.

[8] F. Akopyan *et al.*, “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 34, no. 10, pp. 1537–1557, 2015.

[9] Q. Chen, G. He, X. Wang, J. Xu, S. Shen, H. Chen, Y. Fu, and L. Li, “A 67.5μJ/prediction accelerator for spiking neural networks in image segmentation,” *IEEE Trans. Circuits Syst. II*, pp. 1–1, 2021.

[10] Q. Chen, C. Gao, and Y. Fu, “Cerebron: A reconfigurable architecture for spatiotemporal sparse spiking neural networks,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 10, pp. 1425–1437, 2022.

[11] A. Khodamoradi, K. Denolf, and R. Kastner, “S2N2: A fpga accelerator for streaming spiking neural networks,” in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*.

[12] H. Asgari, B. M.-N. Maybodi, M. Payvand, and M. R. Azghadi, “Low-energy and fast spiking neural network for context-dependent learning on fpga,” *IEEE Trans. Circuits Syst. II*, vol. 67, no. 11, pp. 2697–2701, 2020.

[13] D. Neil and S.-C. Liu, “Minitaur, an event-driven fpga-based spiking network accelerator,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 22, no. 12, pp. 2621–2628, 2014.

[14] B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, and S. Liu, “Conversion of continuous-valued deep networks to efficient event-driven networks for image classification,” *Front. Neurosci.*, vol. 11, pp. 1–12, 2017.

[15] S. Li, Z. Zhang, R. Mao, J. Xiao, L. Chang, and J. Zhou, “A fast and energy-efficient snn processor with adaptive clock/event-driven computation scheme and online learning,” *IEEE Trans. Circuits Syst. I*, vol. 68, no. 4, pp. 1543–1552, 2021.

[16] H. Fang, Z. Mei, A. Shrestha, Z. Zhao, Y. Li, and Q. Qiu, “Encoding, model, and architecture: Systematic optimization for spiking neural network in fpgas,” in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[17] J. Zhang, H. Wu, J. Wei, S. Wei, and H. Chen, “An asynchronous reconfigurable snn accelerator with event-driven time step update,” in *2019 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 2019, pp. 213–216.